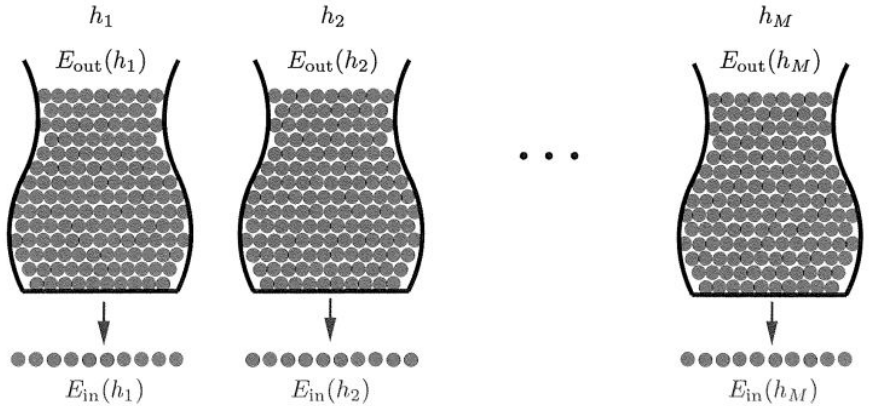
EP1

Simulação da Vida, do Universo e tudo mais

Proposta

Simular a diferença entre erro in e erro out a partir de um universo (pré-estabelecido) e um conjunto de hipóteses bem definido e finito!



$$\begin{aligned}\mathbb{P}[\,|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\,] \;\leq\; \mathbb{P}[\;& |E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon \\ & \textbf{or}\ |E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon \\ & \cdots \\ & \textbf{or}\ |E_{\text{in}}(h_M) - E_{\text{out}}(h_M)| > \epsilon\,] \\ \leq\; & \sum_{m=1}^{M} \mathbb{P}\left[|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon\right].\end{aligned}$$

Proposta

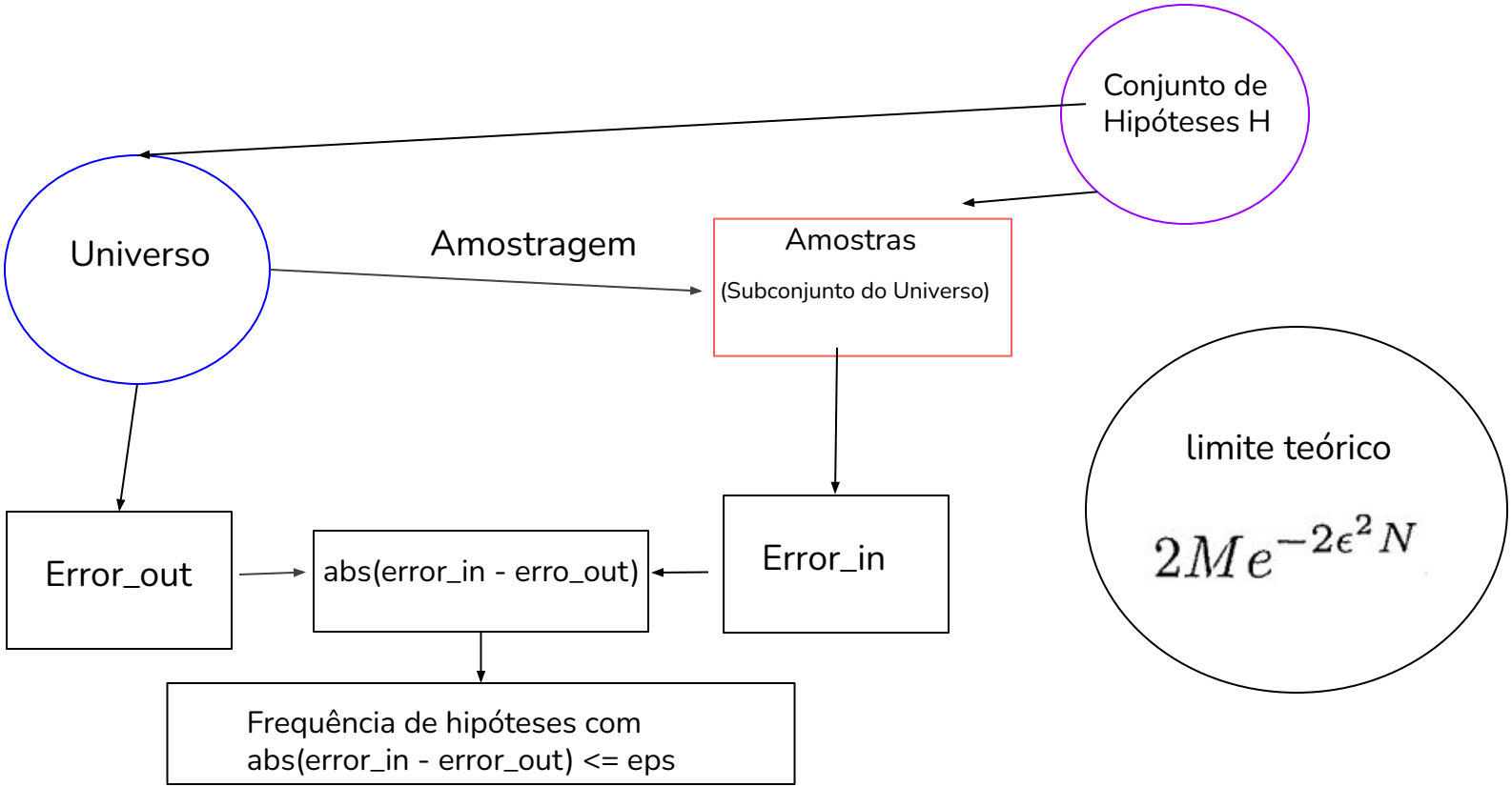
Verificar validade do teorema 1: $\mathbb{P}[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}.$

that assures us that $E_{\text{out}}(g) \approx E_{\text{in}}(g)$ so we can use E_{in} as a proxy for E_{out}.

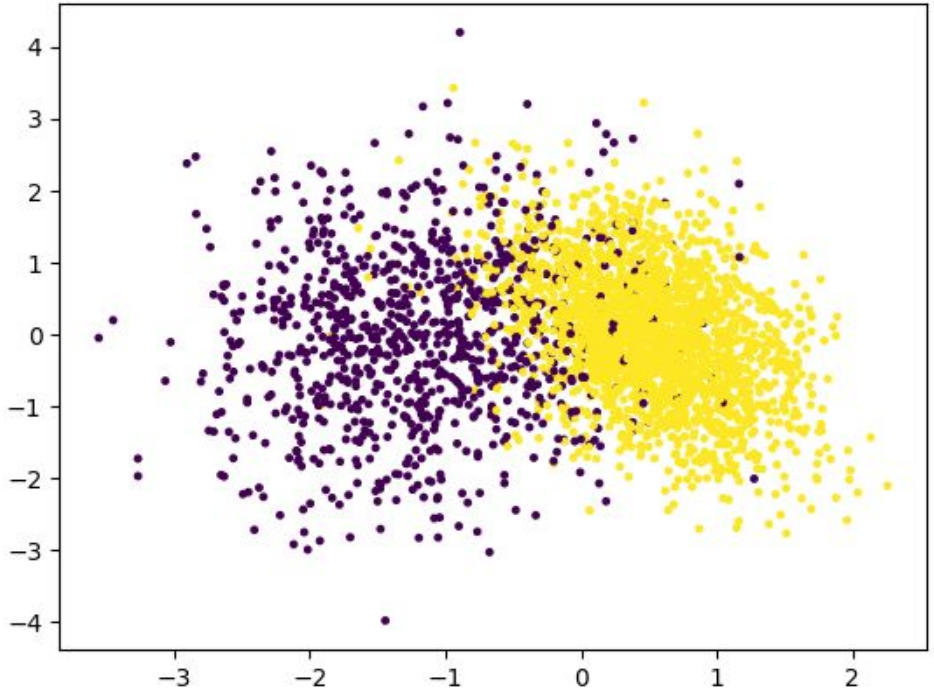
Comparando o limite teórico dado pelo teorema 1 e a probabilidade obtida através da aproximação pela frequência do número de ocorrências de hipóteses com diferença entre o valor absoluto do error_in e erro_out ser menor ou igual a um dado valor de epsilon.

1. Can we make sure that $E_{\text{out}}(g)$ is close enough to $E_{\text{in}}(g)$?
2. Can we make $E_{\text{in}}(g)$ small enough?

Proposta


Conjunto de Hipóteses H
Universo
Amostragem
Amostras
(Subconjunto do Universo)
limite teórico
$2Me^{-2\epsilon^2 N}$
Error_out
abs(error_in - erro_out)
Error_in
Frequência de hipóteses com
abs(error_in - error_out) <= eps

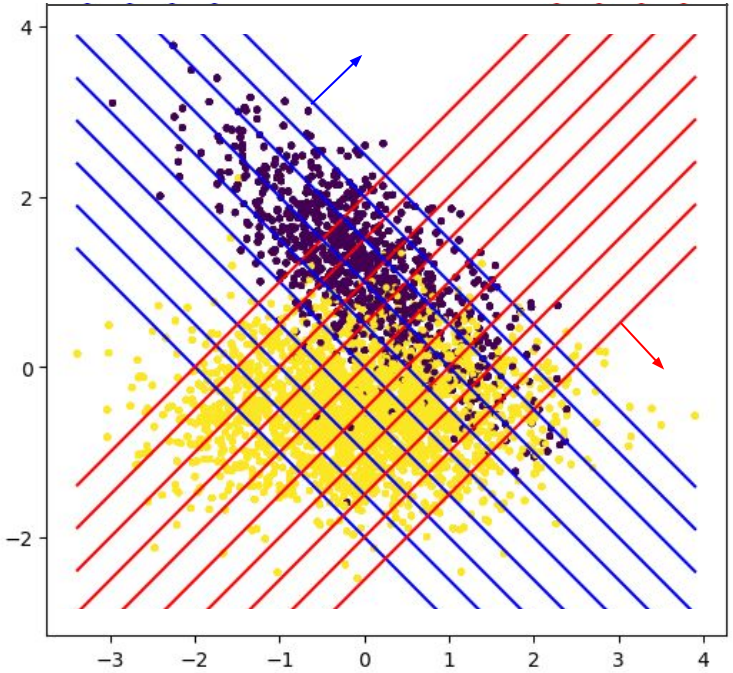
Dataset (X e Y)



O universo consistirá de uma grande quantidade de pontos em R^2 com as respectivas classes conhecidas {-1, +1}

Obs: A avaliação gerará aleatoriamente um dataset para testar as funções

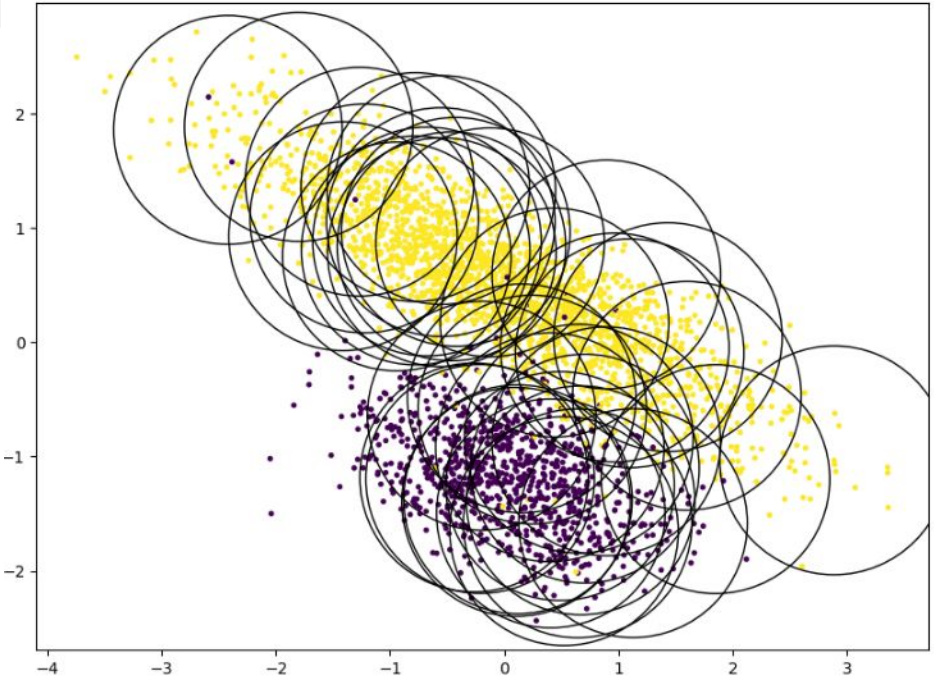
Conjunto de hipóteses (H)



Hipótese 1:

Perceptrons diagonais

Conjunto de hipóteses (H)



Hipótese 2:

Circunferências

standardization (valor: 0.5)

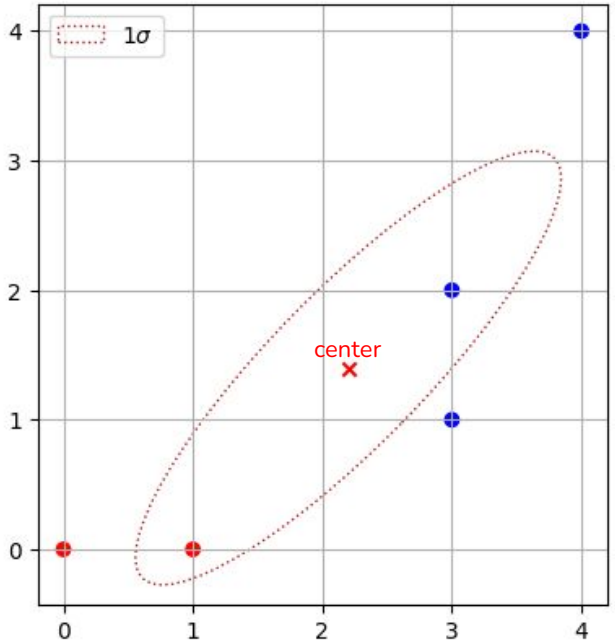
```
def standardization(X):
    '''
      Given a np.array X returns X_std: mean = 0, std = 1 (not inplace - pure function)
    '''
```

No pré processamento dos dados em ML, costumamos padronizar os dados. Isto é, fazer uma transformação linear para que o centro fique em zero e o desvio padrão 1.

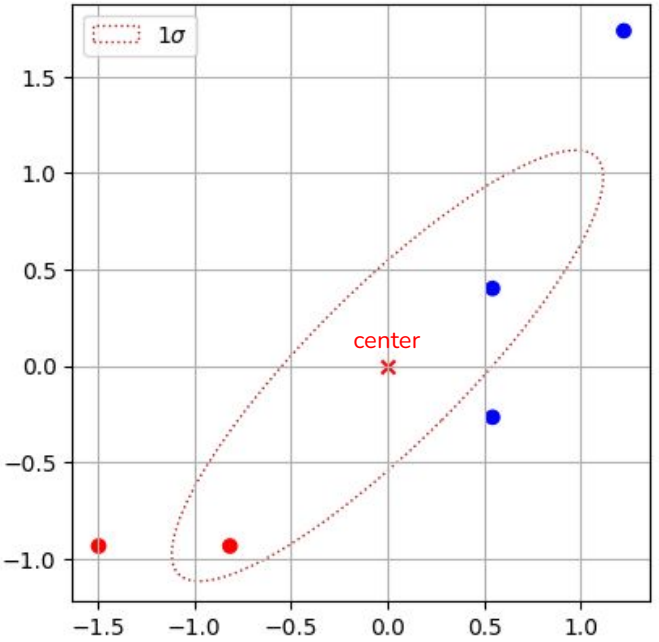
Repare que o Universo deve ser padronizado, mas depois, quando se fizer uma amostragem, a amostra também deverá ser padronizada, mesmo que isso já tenha sido feito com o Universo uma vez que a média e o desvio padrão da amostra podem não ser os mesmos valores do Universo.

standardization (antes e depois)

```
X = np.asarray([[0,0],[1,0],
                [3,1],[3,2],[4,4]])
```




```
X_std=standardization(X)
```



calc_error (valor: 1.0)

```
def calc_error(Y,Y_hat):
  '''
    Given Y (labels) and Y_hat(predicts), returns normalized error
    Inputs:
      Y: np.array or list
      Y_hat: np.array or list
  '''
```

Será utilizada tanto para calcular E_{in} e E_{out}.

O erro deverá ser normalizado pelo número de elementos (do Universo ou da amostra)

Para as listas:

a = np.zeros(5) e b = np.ones(5)

calc_error(a,b) = 1 , calc_error(a,a) = 0

Para as listas:

y = [1,1,1,-1,-1] Y_hat = [-1,1,-1,1,-1]

calc_error(y,y_hat) = 0.6

sampling (valor: 0.5)

```
def sampling(N,X,Y,random_state=42):
  '''
    Given the N #of samples (to sampling), X (np.array) and Y (labels - np.array)
    returns N random samples (X,y) type: np.array
  '''
```

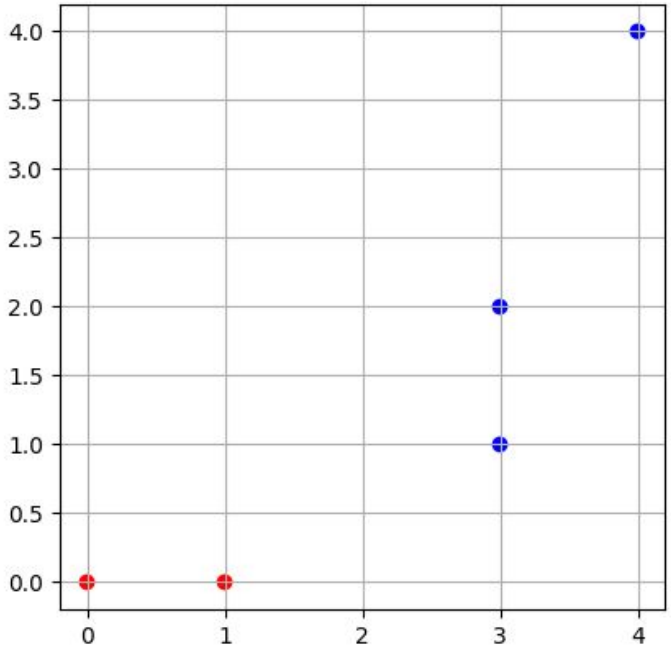
Será utilizada para pegar uma amostra do Universo X.

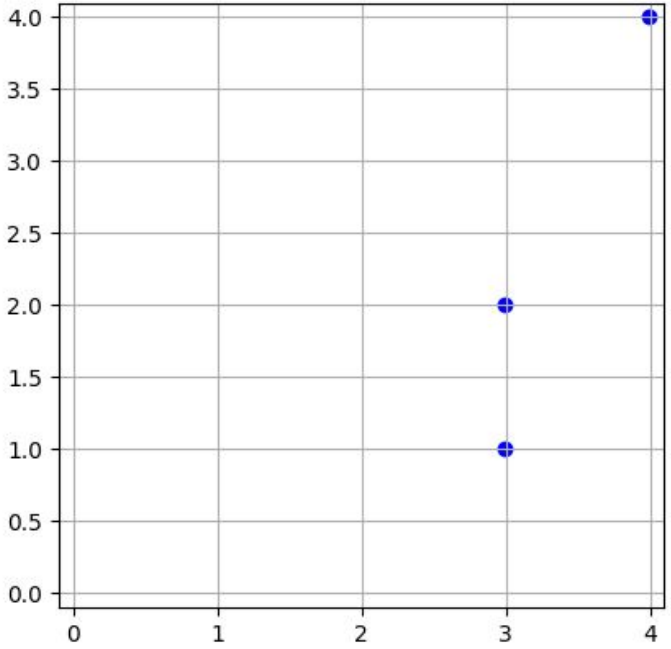
Utilize a função randint do numpy.

sampling

```
X = np.asarray([[0,0],[1,0],
                [3,1],[3,2],[4,4]])
```

```
X_sample,y_sample=f.sampling(3,X,Y,42)
```





diagonais (valor: 2.5)

```
def diagonais(X,M,b):
  '''
Função Diagonais: retas 45º (coeficiente angular +1 e -1  variando bias
um tanto para frente e um tanto para trás - passo do bias (b passado por parâmetro)
definido pelo intervalo [-M//4,M//4)

Sabendo que:
  x0 * w[0] + x1 * w[1] + bias = 0 e que
  w = [1,1] no caso da reta com inclinação negativa e
  w = [1,-1] no caso da reta com inclinação positiva

A seguinte ordem deve ser utilizada:
  bias partindo de -(M//4) * b até (M//4) * b (exclusive)
  A reta com inclinação negativa (coef == -1), vetor w = [1,1] (perpendicular a reta), e bias é calculda primeiro
  e a na sequência reta com inclinação positiva, vetor w = [1,-1], e o mesmo bias.
  Conforme mostrado nos plots!

  parâmetros:
    X: np.array
    M: número de hipóteses do universo (número inteiro) - espera-se um múltiplo de 4
  Retorna
    predict: np.array de np.array de y_hat, um y_hat para cada hipótese (reta), deve ter tamanho M
'''
```

diagonais

[-M//4, (-M//4 + 1), ... ,M//4)

b : passado por parâmetro na função

reta com inclinação negativa, w = [1, 1], e um bias dado pela combinação acima

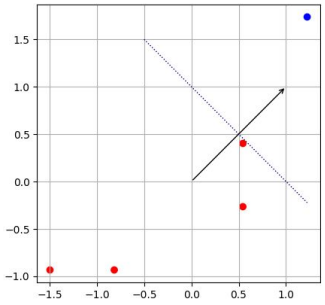
reta com inclinação positiva, w = [1, -1], e um bias dado pela combinação acima

A função retorna uma lista de listas com valores {-1 , +1} de acordo com a classificação dos pontos para cada perceptron

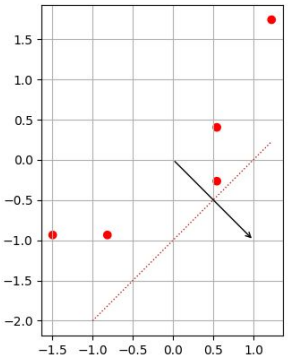
Prestar muita atenção ordem (predicts por hipótese)

diagonais

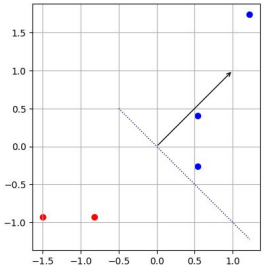
```
predicts=diagonais(X_std,4,1)
```



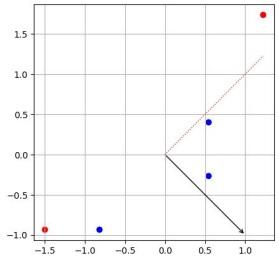
predicts[0]



predicts[1]



predicts[2]



predicts[3]

Repare que para o primeiro e segundo caso o valor do parâmetro b é o mesmo!
Mas a reta cruza o eixo y em pontos diferentes, porque a equação da reta é dado por:
x0 * w[0] + x1 * w[1] + bias = 0
Tal diferença ocorre já que no caso 1 w = [1,1] e no caso 2 w = [1,-1]

euclidean_dist (valor: 0.5)

```
def euclidean_dist(p,q):
    '''
      Given two points (np.array) returns the euclidean distance between them
    '''
```

É utilizada para verificar se a predição é positiva ou negativa no caso da hipótese <egocentric>.

Distância menor que o raio será classe positiva, caso contrário, negativa.

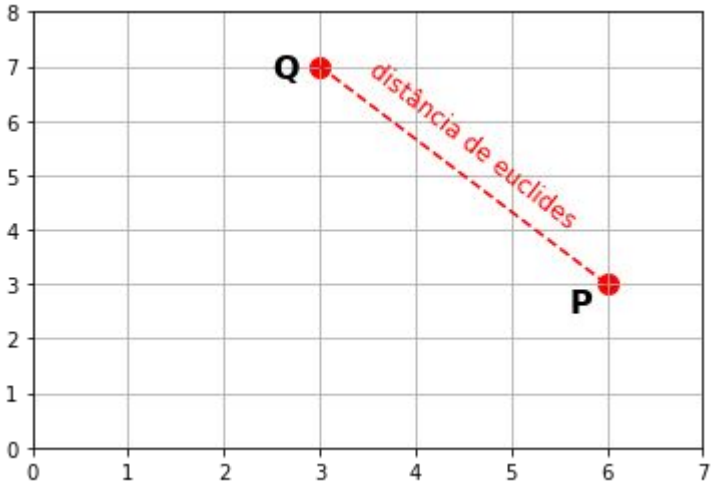
euclidean_dist

$$D = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

```
P = [6.0, 3.0]
Q = [3.0, 7.0]
print(euclidean_dist(P, Q))
```

Saída:

5.0



egocentric (valor: 2.0)

```
def egocentric(X,C,r):
  '''
    Given a dataset X (np.array), C (np.array) are the points that will be used as centers, and a radius r:
      For each point in C, Creates a circumference c, each center works as an hypothesis, and classify points inside c as +1
      otherwise -1.
      Returns all predicts (an list for each point (used as center) )
  '''
```

Conjunto de hipóteses de circunferências

egocentric

X: pontos que serão classificados

C: (subconjunto de X) pontos utilizados como centros de hipóteses (circunferências)

utilizaremos o conjunto amostrado! - portanto o número de hipóteses será N

r: raio da circunferência da hipótese

egocentric

Para os valores:

```
X = np.asarray([[0,0],[1,0],[3,1],[3,2],[4,4]])

Y = np.asarray([-1,-1,1,1,1])

N = 2
```

Após as padronizações...

A chamada egocentric(X_std,X_sampled_std,1) retorna:

```
[[1, 1, -1, -1, -1], [-1, -1, -1, 1, 1]]
```

calc_freq (valor: 3.0)

```
def calc_freq(N,H_set,eps,X,Y,M=100,b=0.05,r=1,random_state = 42):
  '''
  Given N # of samples(integer), H_set name of the hypotheses set
  (string <diagonais> or <egocentric> error will be returned otherwise)
  eps: epsilon (abs(error_in - error_out) desired), X from the Universe data (np.array - complete dataset),
  Y is all label from theentire Universe(np.array), M # of hypotheses used if <diagonais> is chosen,
  B: is the bias used when <diagonais> is chosen, r radius of the circumference if <egocentric> is chosen,
  random_state to set the seed

  Returns:
    bound: theoretical bound for Pr[abs(error_in - error_out) > eps]
    probs: approximated probability of Pr[abs(error_in - error_out) <= eps] by the frequency
      (# of occurancies (abs(error_in - error_out) <= eps) / # of hipotheses)
  '''
```

calc_freq

Atenção com a padronização dos dados:

fazer separado para dados amostrados e para o universo

Atenção com o cálculo dos limites teóricos (depende do H):

diagonais: # hipóteses é dado pelo # de retas

egocentric: # hipóteses é dado pelo tamanho da lista C

probs: # ocorrências da diff(error_in - error_out) <= eps / # of h

calc_freq

Utilizando o dataset do slide “Apendicite”, e os valores:

`eps = 0.1, N = 700, M = 500, b = 1`

A chamada `calc_freq(N,"diagonais",eps,X,Y,M,b)`

retorna: `(0.0008315287191035649, 1.0)`

Já para os valores:

`eps = 0.1 , N = 700`

A chamada `calc_freq(N,"egocentric",eps,X,Y)`

retorna: `(0.0011641402067449908, 1.0)`

Outras funções

Você poderá criar outras funções auxiliares, mas elas não serão consideradas para fins de avaliação.

O arquivo EP1.py será importado como módulo para ter as funções avaliadas. Portanto, não deverá conter linhas de código fora das funções.

Se quiser testar, teste no colab ou crie uma função main() e não apague as linhas que estão no final do template

```
if __name__ == "__main__":
    main()
```

Considerações importantes:

- Não esqueça de preencher o cabeçalho com seu nome e número USP, se não é zero.
- Plágio, se for detectado, é zero.

O termo *plágio*++ é usado no contexto deste curso para indicar uma série de comportamentos **ligados** à **apresentação de trabalho individual com indicação falsa de individualidade ou autoria**.

Existem várias definições do termo *plágio* em dicionários e em textos legais. Plágio++ é uma versão incrementada desse conceito que é usada nos cursos de computação, no contexto dos trabalhos ou provas apresentados no curso. Isto quer dizer que: tudo que for plágio pela definição legal, é plágio++, mas plágio++ engloba algumas ações que podem não ser consideradas plágio no sentido legal, e aqui são tratadas severamente.

A principal diferença entre plágio++ e simples plágio, é que quando são detectados trabalhos ou provas em que houve cópia, **ambos** são tratados como plágio++; a autoria real é irrelevante. Além disso, "cópia disfarçada" também é cópia.

Mais sobre plágio

Existem algumas razões para o tratamento rigoroso do plágio++, entre as quais:

- O plágio é uma ação anti-ética e ilegal, quando executado à revelia do autor.
- Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação. Dependendo da área do estudante, a nota será parte da classificação usada para entrada em sub-áreas; quem falsifica as notas está enganando o sistema e seus colegas.
- Um dos objetivos do "curso de MAC" é o aprendizado individual dos conceitos básicos de programação. A principal evidência desse aprendizado está nos EPs, os exercícios-programa, que necessariamente são individuais neste curso.
- A detecção de plágio++ é parte das atividade usuais de avaliação dos trabalhos ou provas e atribuição de notas, que seguem o ritmo do calendário do curso.

Continuação importante

O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar como resolveu um ou outro problema. Por exemplo, pode explicar que para fazer as contas pedidas no enunciado é preciso usar dois loops ou que para representar os dados basta usar algumas variáveis etc... O que você não deve fazer é mostrar o seu código! Você pode achar que a amizade é mais importante do que as considerações éticas acima, mas mostrar código pode prejudicar o aprendizado do seu colega:

- depois de o seu colega ter visto o seu código, será muito mais difícil para ele imaginar uma solução original;
- o seu colega não entenderá realmente o problema: a compreensão passa pela feitura dos EPs e não pela imitação/cópia;
- é impressionante como são diferentes as provas escritas por alunos que fizeram EPs e pelos que não fizeram, e as notas baixas desses últimos são inevitáveis.

Além disso, um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela.

Esses problemas ocorrem em muitos lugares, as políticas e recomendações não variam muito.

Mais continuação importante

- A entrega do EP deverá ser feita na tarefa correspondente do e-disciplinas.
- Não serão aceitos arquivos entregues por e-mail, papel, caderno, carta, fax...
- Prazo de entrega - seja organizado pois não serão aceitas entregas fora do prazo.
- Não pense que não conferimos os códigos porque possuem a correção automática

Estas funções do EP são úteis para que você possa fazer testes para verificar o comportamento de hipóteses e datasets. Depois de criadas, o ideal é que você faça seus próprios testes, plotar os resultados, explorar um pouco mais o que foi apresentado na disciplina. Não se contente só com tirar dez porque o código rodou.

Apendicite

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
```

Dataset

```
[ ]  1 seed=42
```

```
[ ]  1 X, Y = make_classification(n_samples = 3000, n_classes=2,
     2     n_features=2, n_redundant=0, n_informative=2, n_clusters_per_class=1,
     3     n_repeated=0, hypercube=True, weights=[0.3,], random_state=seed
     4 )
```

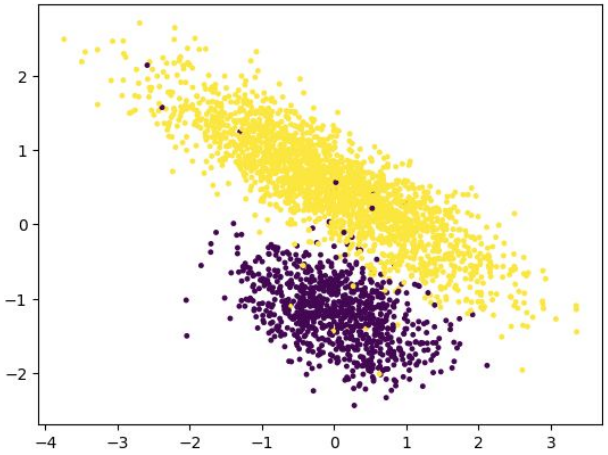
```
[ ]  1 Y
```

```
array([1, 1, 1, ..., 1, 0, 1])
```

```
[ ]  1 Y[Y == 0] = -1
```

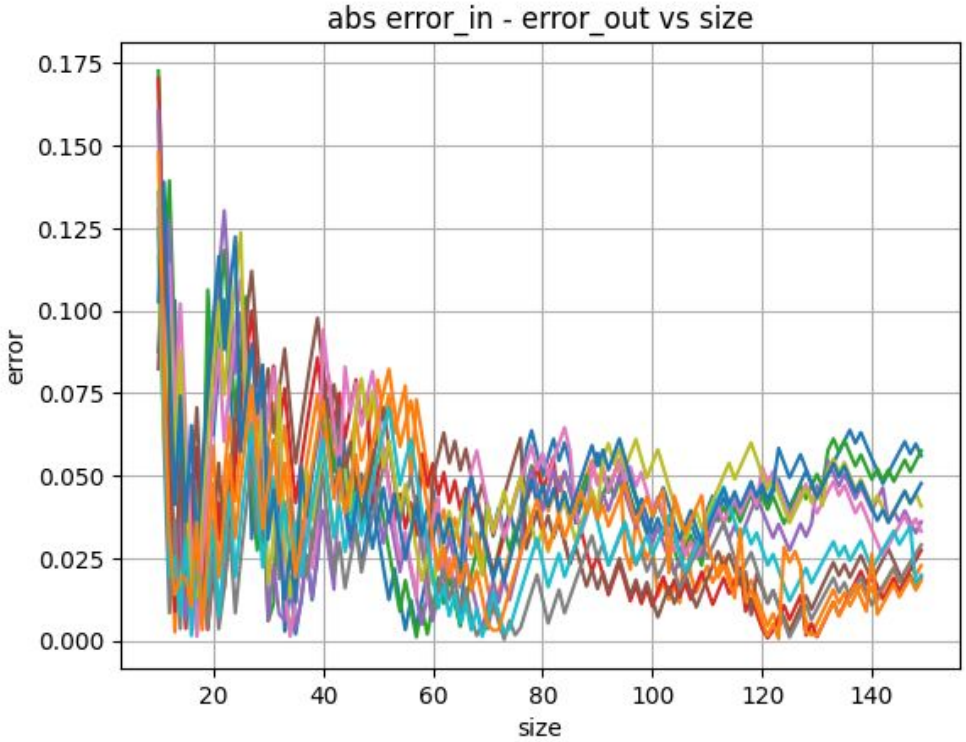
```
[ ]  1 Y
```

```
array([ 1,  1,  1, ...,  1, -1,  1])
```




```
plt.scatter(X_std[:, 0], X_std[:, 1], marker=".", c=Y, s=25)
```

Extra:



Plotar o gráfico de $|E_{in}\text{-}E_{out}|$ variando o tamanho da amostra N